Notes for Tuesday March 3, 1998

1 Overview

These notes are an annotated listing of the kinds of commands you would use to fit log-linear models to tables of counts using `glm()` in SPLUS. They do not represent complete, careful data analyses. In addition to these notes, I have handed out Xeroxed copies of Chapter 6 of Chambers and Hastie (1992), which is somewhat more complete.

2 Example

Consider the following data (from Problem 5.3 in Fienberg, 1987). In a study of the relationship between car size and accident injuries, accidents were classified according to the type of accident, severity of accident, and whether or not the driver was ejected. The data are presented below.

		Accident Type			
		Collision		Rollover	
		Severity		Severity	
Car weight	Driver ejected	Not severe	Severe	Not severe	Severe
Small	No	350	150	60	112
	Yes	26	23	19	80
Standard	No	1878	1022	148	404
	Yes	111	161	22	265

Our mission is to find an appropriate model for these data, but in order to do that we need to;

- Enter the data into Splus and organize it like a table,
- Find a way of looking at the data,
- Coerce Splus into fitting some appropriate loglinear models,
- Look at some diagnostic measures.

2.1 Entering the Data

Assume that data is in a nice simple format, with only the numbers and not the cell labels. This is most common. Here is the data, as stored in the file called `glm.dat`.

```
350    150   60  112
26      23   19   80
1878  1022  148  404
111    161   22  265
```

Just read this as a list of numbers into Splus,

```
402 > accidents <- scan("glm.dat")
```

Now you need to figure out the "design" of the data—how to associate different levels of the factors with the various counts. Splus has a useful command called `fac.design`, that we have used before, which will generate factorial designs. Here is an example of using it. For documentation use Splus's help facility.

```
402 > fac.design(c(2,2,2,2),factor.names=c("Type","Sev","Eject","Weight"))
    Type  Sev Eject Weight
 1 Type1 Sev1 Ejct1  Wght1
 2 Type2 Sev1 Ejct1  Wght1
 3 Type1 Sev2 Ejct1  Wght1
 4 Type2 Sev2 Ejct1  Wght1
 5 Type1 Sev1 Ejct2  Wght1
 6 Type2 Sev1 Ejct2  Wght1
 7 Type1 Sev2 Ejct2  Wght1
 8 Type2 Sev2 Ejct2  Wght1
 9 Type1 Sev1 Ejct1  Wght2
10 Type2 Sev1 Ejct1  Wght2
11 Type1 Sev2 Ejct1  Wght2
12 Type2 Sev2 Ejct1  Wght2
13 Type1 Sev1 Ejct2  Wght2
14 Type2 Sev1 Ejct2  Wght2
15 Type1 Sev2 Ejct2  Wght2
16 Type2 Sev2 Ejct2  Wght2
```

Now all we need to do is marry the data with the design in a data frame,

```
402 > accidents _ data.frame(Resp=accidents,fac.design(c(2,2,2,2),
+  factor.names=c("Type","Sev","Eject","Weight")))

402 > accidents
   Resp  Type  Sev Eject Weight
 1  350 Type1 Sev1 Ejct1  Wght1
 2  150 Type2 Sev1 Ejct1  Wght1
 3   60 Type1 Sev2 Ejct1  Wght1
 4  112 Type2 Sev2 Ejct1  Wght1
 5   26 Type1 Sev1 Ejct2  Wght1
 6   23 Type2 Sev1 Ejct2  Wght1
 7   19 Type1 Sev2 Ejct2  Wght1
 8   80 Type2 Sev2 Ejct2  Wght1
 9 1878 Type1 Sev1 Ejct1  Wght2
10 1022 Type2 Sev1 Ejct1  Wght2
11  148 Type1 Sev2 Ejct1  Wght2
12  404 Type2 Sev2 Ejct1  Wght2
```

```
13  111 Type1 Sev1 Ejct2  Wght2
14  161 Type2 Sev1 Ejct2  Wght2
15   22 Type1 Sev2 Ejct2  Wght2
16  265 Type2 Sev2 Ejct2  Wght2
```

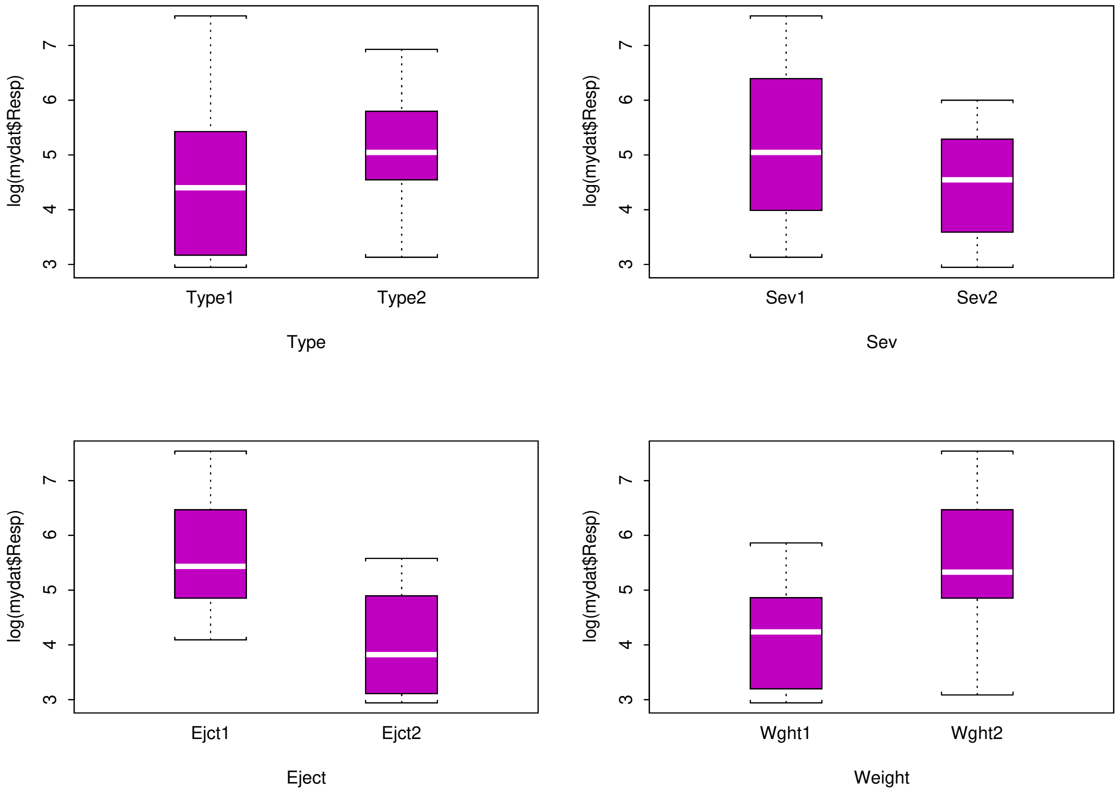
As you can see, this generated the correct design. Using the **fac.design** command makes sure that the classifying variables really are treated as Splus factors.

2.2 Looking at the data

For most tables of counts we will build a log-linear model using the Poisson or Multinomial sampling model (either sampling model will do).

Since a log-linear model is an ANOVA-style linear model on the scale of $\log(x_{ijk})$, we can look at the log-counts using the same kind of exploratory plots that we used for ANOVA models.

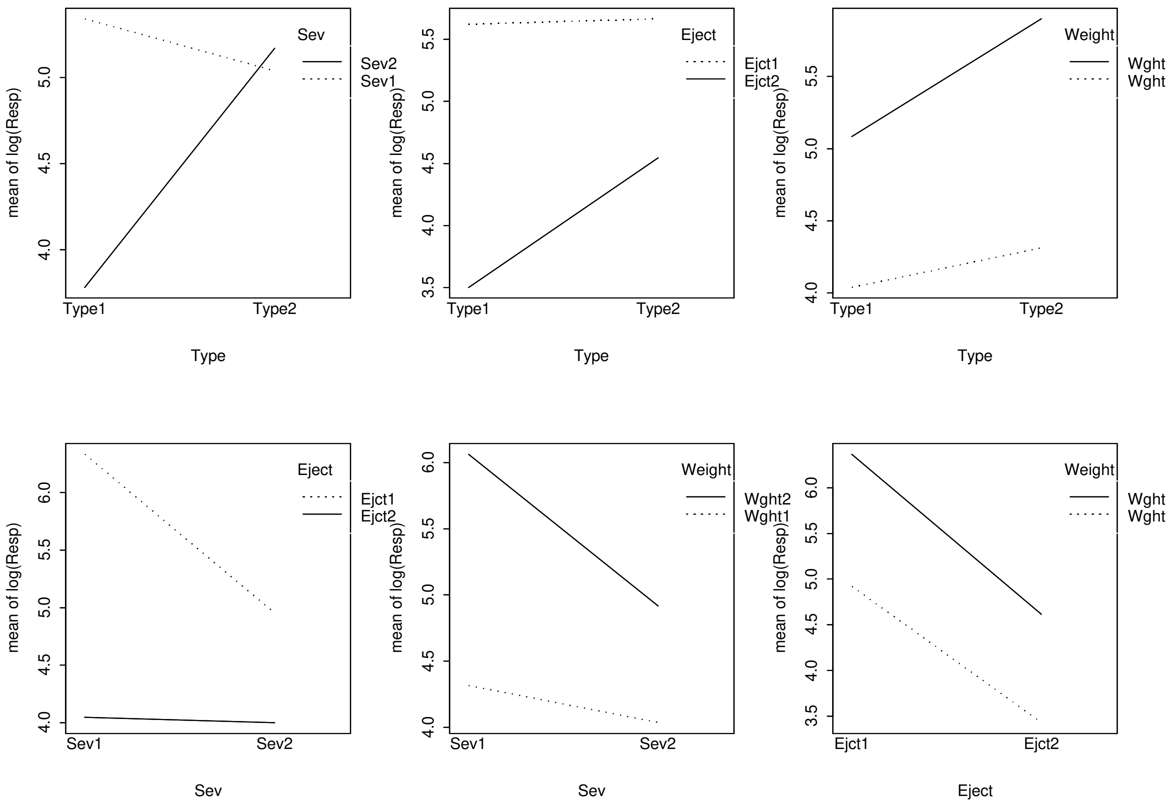
```
402 > par(mfrow=c(2,2))
402 > plot.factor(accidents,log(accidents$Resp),ask=F)
```



The boxplots suggest that we will find significant main effects for eject and weight, less clearly so for Type and Severity.

```
402 > names(accidents)
```

```
[1] "Resp"   "Type"   "Sev"    "Eject"  "Weight"
402 > attach(accidents)
402 > par(mfrow=c(2,3))
402 > interaction.plot(Type,Sev,log(Resp))
402 > interaction.plot(Type,Eject,log(Resp))
402 > interaction.plot(Type,Weight,log(Resp))
402 > interaction.plot(Sev,Eject,log(Resp))
402 > interaction.plot(Sev,Weight,log(Resp))
402 > interaction.plot(Eject,Weight,log(Resp))
402 > detach()
```



The strongest case for an interaction is with Type and Severity; though some of the other plots involving Severity look like at least candidates for interactions as well.

2.3 Fitting Models

This is the fun part. The Splus `glm` command allows you to fit all sorts of hierarchical loglinear modes. There is also the older `loglin` command which is sometimes useful for very large data sets, but otherwise is slower and less versatile than `glm`.

We might as well start out by fitting just the null model. This is not very interesting, but provides a useful beginning position.

```
402 > acc.glm1 <- glm( Resp ~ 1, family=poisson, data=accidents)
```

This should be read as "The model `acc.glm1` gets the output from the result of fitting a generalized linear model with the formula `Resp ~ 1` (i.e., the *intercept-only* model, also called the *null model*), using a poisson regression model and the default (log) link, and drawing the data from the data frame called `acc.glm1`."

There are now many questions one can ask of the resulting `acc.glm1`, including questions about the deviance, residuals, fitted and estimated values, etc. For example the deviance (or likelihood ratio goodness of fit statistic, or G^2) is

```
402 > deviance(acc.glm1)
[1] 7686.358
```

and a summary of the fitted model is,

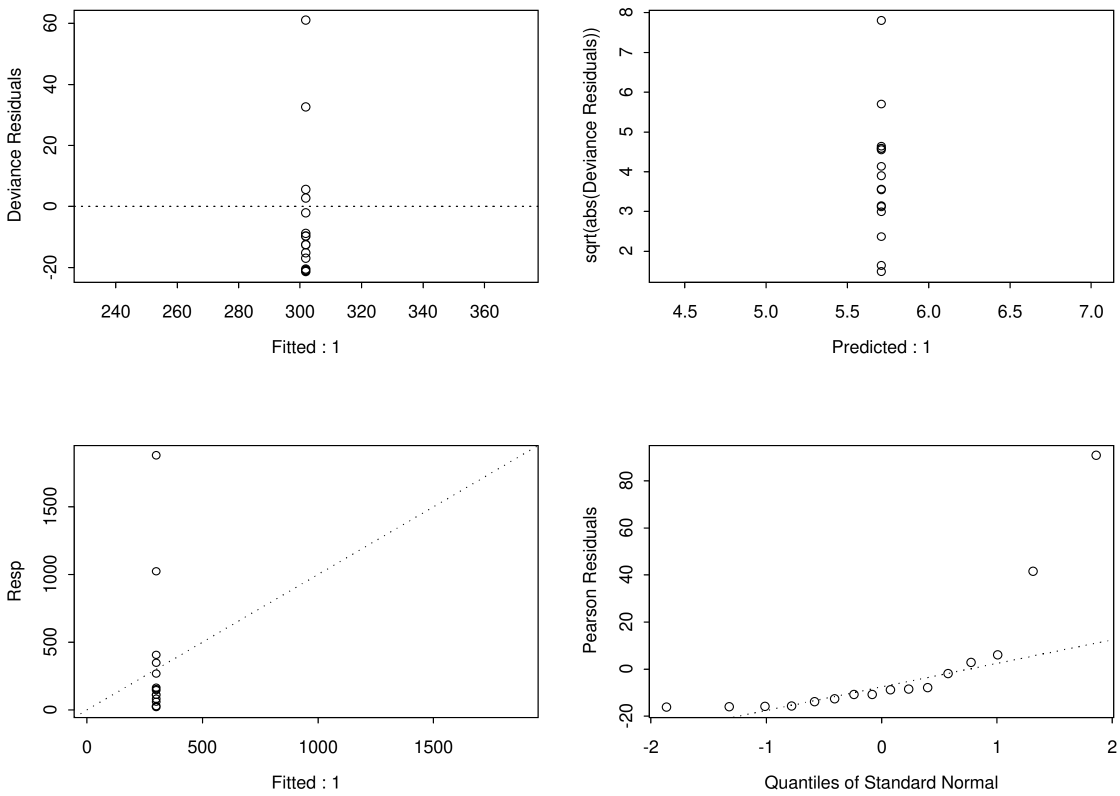
```
402 > summary(acc.glm1)

Call: glm(formula = Resp ~ 1, family = poisson, data = accidents)
Deviance Residuals:
       Min        1Q   Median         3Q      Max
 -21.46568 -17.92141 -11.1995 -0.9543187 60.93338

Coefficients:
              Value Std. Error  t value
(Intercept) 5.71022   0.404268 14.12484

(Dispersion Parameter for Gaussian family taken to be 789.5429 )

    Null Deviance: 7686.358 on 15 degrees of freedom

Residual Deviance: 7686.358 on 15 degrees of freedom

Number of Fisher Scoring Iterations: 0
```

We can verify that SPLUS used the Poisson sampling model and log link function using the `family()` function,

```
402 > family(acc.glm1)
Poisson family

    link function:   Log: log(mu)
variance function:   Identity: mu
```

and we can get diagnostic plots as follows:

```
402 > par(mfrow=c(2,2))
402 > plot(acc.glm1)
```



These plots are largely uninteresting for the null model, since there's only one prediced or fitted value for the model. However the normal probability plot for the Pearson's residuals does suggest some model misfit here.

We can manually fit alternate models, either by entering a completely new model via `glm`, or more handily using the `update` function, like,

```
402 > acc.glm2 <- update(acc.glm1, . ~ Type + Sev + Eject + Weight)
```

which should be read as "The new value of the model structure `acc.glm2` should be updated from the old value by changing the model formula, but leaving the data and class of model (poisson) the same. The new model formula has the same left hand side as the old one, but the right hand side is now `Type + Sev + Eject + Weight`, which represents the complete independence model."

The same summaries as for linear models can now be produced, such as, the analysis of deviance table,

```
402 > anova(acc.glm2)
Analysis of Deviance Table

Poisson model

Response: Resp
```

```
Terms added sequentially (first to last)
       Df Deviance Resid. Df Resid. Dev
  NULL                    15   7686.358
  Type  1   32.661        14   7653.697
   Sev  1 1489.432        13   6164.265
 Eject  1 2674.713        12   3489.551
Weight  1 2296.445        11   1193.106
```

and the `anova()` command can also be used to construct specific tests, e.g.

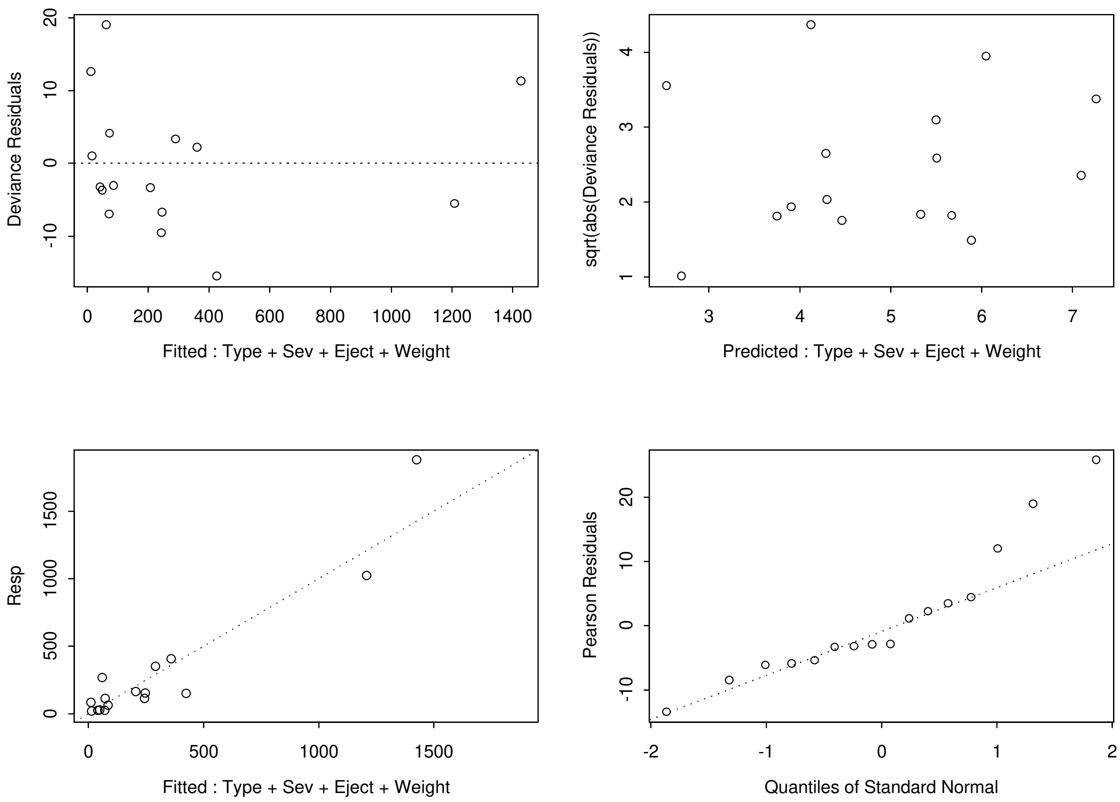
```
402 > anova(acc.glm2,update(acc.glm2, . ~ . - Weight))
Analysis of Deviance Table

Response: Resp

                        Terms Resid. Df Resid. Dev    Test Df  Deviance
1 Type + Sev + Eject + Weight        11   1193.106
2          Type + Sev + Eject        12   3489.551 -Weight -1 -2296.445
```

similarly, tests can be constructed for adding terms to a model.

There is also a `lm()`-style summary, including parameter estimates,

```
402 > summary(acc.glm2)

Call: glm(formula = Resp ~ Type + Sev + Eject + Weight, family = poisson, data =
        accidents)
Deviance Residuals:
       Min        1Q    Median       3Q      Max
 -15.57728 -5.843355 -3.169368 3.511405 19.09357

Coefficients:
                  Value Std. Error     t value
(Intercept)  4.90063958 0.02567420  190.877988
       Type -0.08236347 0.01442982   -5.707864
        Sev -0.60481489 0.01708364  -35.403166
      Eject -0.88176920 0.02031366  -43.407704
     Weight  0.79374577 0.01915400   41.440201

(Dispersion Parameter for Poisson family taken to be 1 )

    Null Deviance: 7686.358 on 15 degrees of freedom

Residual Deviance: 1193.106 on 11 degrees of freedom

Number of Fisher Scoring Iterations: 4
```

```
Correlation of Coefficients:
       (Intercept)       Type        Sev      Eject
  Type  0.0462656
   Sev  0.3591861   0.0000000
 Eject  0.5588092   0.0000000  0.0000000
Weight -0.4927811   0.0000000  0.0000000  0.0000000
402 > par(mfrow=c(2,2))
402 > plot(acc.glm2)
```



These residual plots look better (and more familiar) however the normal probability plot still suggests some problems with the fit.

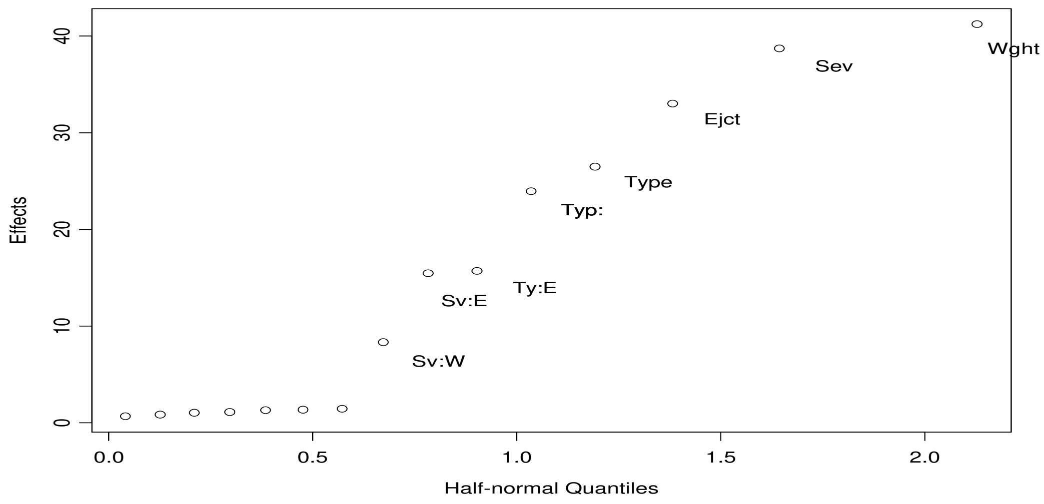
And finally let's fit the full model, with all interactions present.

```
402 > acc.glm3 _ glm(Resp ~ Type*Sev*Eject*Weight, family=poisson, data=accidents)
402 > summary(acc.glm3)

Call: glm(formula = Resp ~ Type * Sev * Eject * Weight, family = poisson, data =
accidents)

Coefficients:
                            Value Std. Error      t value
          (Intercept)  4.83308010   0.031708  152.4246265
```

```
                 Type  0.27176000   0.031708    8.5707077
                  Sev -0.35699155   0.031708  -11.2587215
                Eject -0.81035970   0.031708  -25.5569476
               Weight  0.65789657   0.031708   20.7485987
             Type:Sev  0.42256899   0.031708   13.3268887
           Type:Eject  0.25018336   0.031708    7.8902281
            Sev:Eject  0.33358062   0.031708   10.5203928
          Type:Weight  0.13527978   0.031708    4.2664242
           Sev:Weight -0.21663528   0.031708   -6.8321961
         Eject:Weight -0.06519026   0.031708   -2.0559563
       Type:Sev:Eject  0.03705641   0.031708    1.1686769
      Type:Sev:Weight  0.04361372   0.031708    1.3754801
    Type:Eject:Weight  0.05791725   0.031708    1.8265815
     Sev:Eject:Weight -0.03999437   0.031708   -1.2613336
Type:Sev:Eject:Weight  0.02596418   0.031708    0.8188527

(Dispersion Parameter for Poisson family taken to be 1 )

    Null Deviance: 7686.358 on 15 degrees of freedom

Residual Deviance: 0 on 0 degrees of freedom

Number of Fisher Scoring Iterations: 1
```

Just as with factorial designs for ANOVA with one observation per cell, there are no degrees of freedom left over for residual SS's, so we can't directly assess significance of effects and interactions from this model. To help decide which effects are important, we can also make a half-normal plot of the effect sizes.

```
402 > par(mfrow=c(1,1))
402 > qqnorm.aov(acc.glm3,label=T)
```



This suggests that the next model to consider should involve all the main effects, plus two interactions involving Type and two interactions involving Severity. This story is fairly consitent with the earlier interaction plots.

Here we suggest only the anova table for the model with all two-way interactions.

```
402 > anova(glm(Resp ~ (Type + Sev + Eject + Weight)^2,family=poisson,
+ data=accidents))
Analysis of Deviance Table

Poisson model

Response: Resp

Terms added sequentially (first to last)
             Df Deviance Resid. Df Resid. Dev
        NULL                    15   7686.358
        Type  1   32.661        14   7653.697
         Sev  1 1489.432        13   6164.265
       Eject  1 2674.713        12   3489.551
      Weight  1 2296.445        11   1193.106
    Type:Sev  1  601.420        10    591.686
  Type:Eject  1  285.996         9    305.690
 Type:Weight  1    0.757         8    304.933
   Sev:Eject  1  229.333         7     75.600
  Sev:Weight  1   66.580         6      9.020
Eject:Weight  1    1.692         5      7.328
```

2.4 Things still to do

There is a lot more to write here, but this is at least a start. All the standard `lm` things, like stepwise selection, can be done. The `anova()` command can be used to compare models and submodels. Most of the `lm` plots can be done, but not all make complete sense. In addition, visual inspection of the Pearson residuals (as we did last week) can help determine where the model is not fitting well.

Additional illustrations are given in Chapter 5 of Chambers and Hastie (handed out in class last week), Section 7.3 of Venables and Ripley (short but interesting) and Section 8.2 of the SPLUS Guide to Statistics.

In general, as with regression and ANOVA, the goal is to find the most parsimonious (= simple) model that is not contradicted by the data, and then to interpret the fitted model in terms of the situation that generated the data.